ADO.NET Overview

ADO.NET is the .NET programming environment for building database applications based on native database formats or XML data. ADO.NET is designed as a back-end data store for all .NET programming models, including Web Forms, Web Services, and Windows Forms. Use ADO.NET to manage data in the .NET Framework.

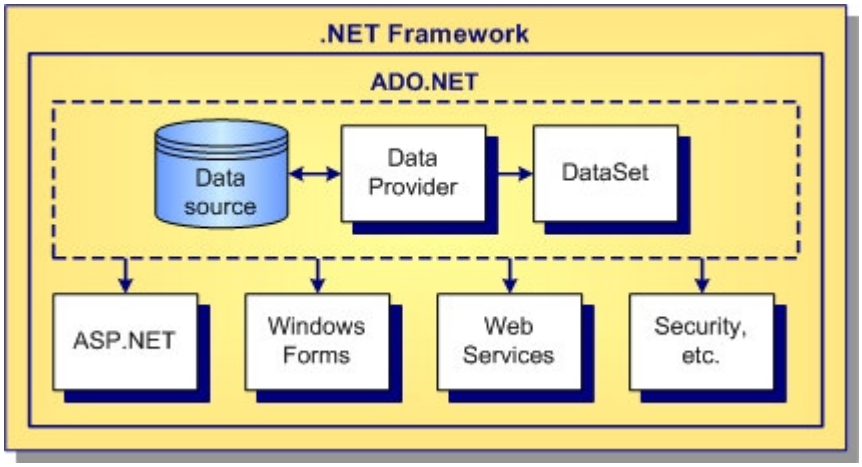
Borland provides tools to simplify rapid ADO.NET development using Borland Data Providers for .NET (BDP.NET). If you are familiar with rapid application development (RAD) and object oriented programming (OOP) using properties, methods, and events, you will find the ADO.NET model for building applications familiar. If you are a traditional database programmer, ADO.NET provides familiar concepts, such as tables, rows, and columns with relational navigation. XML developers will appreciate navigating the same data with nodes, parents, siblings, and children.

This topic discusses the major components of the ADO.NET architecture, how ADO.NET integrates with other programming models in the .NET Framework, and key Delphi 2005 functionality to support ADO.NET.

This topic introduces:

- ADO.NET Architecture
- ADO.NET User Interfaces
- BDP.NET Namespace

ADO.NET Architecture



The two major components of the ADO.NET architecture are the Data Provider and the DataSet. The data source represents the physical database or XML file, the Data Provider makes connections and passes commands, and the DataSet represents one or more data sources in memory. For more information about the general ADO.NET model, see the Microsoft .NET Framework SDK documentation.

Data Source

The data source is the physical database, either local or remote, or an XML file. In traditional database programming, the developer typically works with the data source directly, often requiring complex, proprietary interfaces. With ADO.NET, the database developer works with a set of components to access the data source, to expose data, and to pass commands.

Data Providers

Data Provider components connect to the physical databases or XML files, hiding implementation details. Providers can connect to one or more data sources, pass commands, and expose data to the DataSet.

The .NET Framework includes providers for MS SQL, OLE DB, and Oracle. In addition to supporting the .NET providers, this product includes BDP.NET. BDP.NET connects to a number of industry standard databases, providing a consistent programming environment. For more information, see the Borland Data Providers for Microsoft .NET topic.

The TADONETConnector component provides access to .NET DataSets either directly or through BDP.NET. TADONETConnector is the base class for Delphi 2005 datasets that access their data using ADO.NET. TADONETConnector descendants include TCustomADONETConnector . TADONETConnector is a descendent of TDataSet .

DataSet

The DataSet object represents in-memory tables and relations from one or more data sources. The DataSet provides a temporary work area or virtual scratch pad for manipulating data. ADO.NET applications manipulate tables in memory, not within the physical database. The DataSet provides additional flexibility over direct connections to physical databases. Much like a typical cursor object supported by many database systems, the DataSet can contain multiple DataTables, which are representations of tables or views from any number of data sources. The DataSet works in an asynchronous, non-connected mode, passing update commands through the Provider to the data source at a later time.

Delphi 2005 provides two kinds of DataSets for your use: standard DataSets and typed DataSets. A standard DataSet is the default DataSet that you get when you define a DataSet object implicitly. This type of DataSet is constructed based on the layout of the columns in your data source, as they are returned at runtime based on your Select statement.

Typed DataSets provide more control over the layout of the data you retrieve from a data source. A typed DataSet derives from a DataSet class. The typed DataSet lets you access tables and columns by name rather than collection methods. The typed DataSet feature provides better readability, improved code completion capabilities, and data type enforcement unavailable with standard DataSets. The compiler checks for type mismatches of typed DataSet elements at compile time rather than runtime. When you create a typed dataset, you will see that some new objects are created for you and are accessible through the **Project Manager**. You will notice two files named after your dataset. One file is an XML `.xsd` file and the other is a code file in the language you are using. All of the data about your dataset, including the table and column data from the database connection, is stored in the `.xsd` file. The program code file is created based on the XML in the `.xsd` file. If you want to change the structure of the typed dataset, you can change items in the `.xsd` file. When you recompile, the program code file is regenerated based on the modified XML.

For more information about DataSets, see the Microsoft .NET Framework SDK documentation.

ADO.NET User Interfaces

ADO.NET provides data access for the various programming models in .NET.

Web Forms

Web Forms in ASP.NET provide a convenient interface for accessing databases over the web. ASP.NET uses ADO.NET to handle data access functions.

.NET and BDP.NET connection components ease integration between Web Forms and ADO.NET. DB Web Controls support both ADO.NET and BDP.NET components, accelerating web application development.

Windows Forms

As an alternative to Web Forms, traditional, native-OS clients can function as a front end to ADO.NET databases.

In Delphi 2005 you can provide two types of Windows Forms: a TWinForm object, which is a descendant of TForm and acts as the native .NET Windows Form, and a VCL.NET form.

BDP.NET Namespace

BDP.NET classes are found under the `Borland.Data` namespace.

BDP.NET Namespace

Namespace	Description
Borland.Data.Common	Contains objects common to all Borland Data Providers, including Error and Exceptions classes, data type enumerations, provider options, and Interfaces for building your own Command, Connection, and Cursor classes.
Borland.Data.Provider	Contains key BDP.NET classes like BdpCommand , BdpConnection , BdpDataAdapter , and others that provide the means to interact with external data sources, such as Oracle, DB2, Interbase, and MS SQL Server databases.
Borland.Data.Schema	Contains Interfaces for building your own database schema manipulation classes, as well as a number of types and enumerators that define metadata.

Related Information

- Borland Data Providers for Microsoft .NET
- BDP.NET Data Types
- BDP.NET Component Designers
- Borland DB Web Controls
- Deploying Applications
- Building a Windows Forms Database Application
- Building an ASP.NET Database Application
- Creating and Using Typed DataSets
- Creating Table Mappings
- Microsoft Overview of ADO.NET